

R E M A R K S

Claims 1 through 23 remain in the application. No claim amendments are being made in this paper. Claims 1, 11 and 21 are the independent claims herein. No new matter has been added. Reconsideration and further examination are respectfully requested.

The Examiner evidently has maintained the rejections of the claims based primarily on the Scheuermann reference. In response, applicants will first recap the language of claim 1 and then will point out what applicants believe are errors in the Examiner's position.

Claim 1 is directed to an "apparatus" which includes "an instruction decoder" and "at least one control register coupled to the instruction decoder". The apparatus of claim 1 also includes "an add-compare-select (ACS) engine coupled to the at least one control register". Claim 1 further specifies that "the instruction decoder is operative to control the ACS engine to perform Viterbi decoding in response to the instruction decoder receiving a first instruction, and the instruction decoder is further operative to control the ACS engine to perform turbo decoding in response to the instruction decoder receiving a second instruction".

In maintaining the rejections based on Scheuermann, the Examiner has taken issue with the applicants' point that Scheuermann fails to disclose turbo decoding. In doing so, the Examiner has cited a passage at page 7 of the book by Heegard and Wicker (hereinafter "Heegard book"). Applicants believe the passage cited does not provide a complete definition of "turbo coding" or (more to the point) turbo decoding. Moreover, in focusing only on the presence or absence of turbo decoding in Scheuermann, the Examiner has also failed to consider other deficiencies in the Scheuermann reference.

First, however, applicants will describe how references relating turbo decoding provide a clear definition of turbo decoding that is more detailed than the definition set forth by the Examiner. Moreover, it will be clear that turbo decoding, as properly understood, is not discussed in the Scheuermann reference.

In the passage of the Heegard book cited by the Examiner, it is stated that turbo coding (not decoding) consists of "two key design innovations: parallel concatenated encoding and iterative decoding". However, it is not believed that this definition is complete, nor does it necessarily suffice to distinguish turbo decoding from other types of decoding, including

possibly Viterbi decoding (which is the only type of decoding discussed in Scheuermann). It is believed that a more significant part of the definition of turbo decoding appears in the last paragraph of page 7 of the Heegard book:

The key to the iterative decoder is its exploitation of the component code substructure of the PCE [parallel concatenated encoder]. The iterative decoder has a soft-input/soft-output (SISO) component decoder for each of the component encoders in the PCE. These decoders take turns operating on the received data, forming and exchanging estimates of the message block.

Thus an essential feature of turbo decoding is the use of at least two separate but cooperating decoders. This aspect of turbo decoding is not present in the Scheuermann reference. Accordingly, applicants adhere to their contention that Scheuermann is not concerned in any way with turbo decoding. It is also notable that Scheuermann does not explicitly refer to turbo coding or decoding.

In any event, to substantiate applicants' interpretation of the Heegard book, applicants will now cite further source materials to support their contention that turbo decoding calls for at least two separate but cooperating decoders.

For example, the Wikipedia article entitled "Turbo code" is attached as Exhibit A, and contains the following passage (on page 3 of the attached print-out):

...There are two parallel decoders, one for each of the $n/2$ -bit parity sub-blocks. Both decoders use the sub-block of m likelihoods for the payload data. The decoder working on the second parity sub-block knows the permutation that the coder used for this sub-block.

...

The nitty gritty of turbo codes is how they use the likelihood data to reconcile differences between the two decoders. Each of the two convolutional decoders generates a hypothesis (with derived likelihoods) for the pattern of m bits in the payload sub-block. The hypothesis bit-patterns are compared, and if they differ, the decoders exchange the derived likelihoods they have for each bit in the hypotheses. Each decoder incorporates the derived likelihood estimates from the other decoder to generate a new hypothesis for the bits in the payload. Then they compare these new hypotheses. This iterative process continues until the two decoders come up with the same hypothesis for the m -bit pattern of the payload, typically in 15 to 18 cycles.

Thus this Wikipedia article confirms applicants' interpretation that turbo decoding calls for two separate, cooperating decoders. Further confirmation of applicants' position is found in the article attached hereto as Exhibit B, published Nov. 5, 2005 by Science News Online, entitled "Pushing the Limit: Digital communications experts are zeroing in on the perfect code", authored by Erica Klarreich (hereinafter "Klarreich article"). Applicants believe that the following passage, which appears on page 5 of the attached print-out, is most relevant:

Turbo codes use a divide-and-conquer approach in which each of two decoders gets a different encoded version of the original message and the decoders then collaborate. ...

In the first round, each decoder uses its own clues to guess the bits of the original message, keeping track of how confident it is about each guess. Next, the two decoders compare notes and each updates its guesses and confidence levels. ...

After the two decoders update their proposed solutions, they compare notes again and then update their solutions again. They continue this process until they reach a consensus about the original message. This typically takes 4 to 10 passes of information back and forth.

Like the Wikipedia article, and indeed like the Heegard book, the Klarreich article emphasizes the use of two separate cooperating decoders in turbo decoding.

Still further support for applicants' position is found in the article attached hereto as Exhibit C, which was published March 9, 2005 by IEEE Spectrum Online, entitled "Closing in on the perfect code", authored by Erico Guizzo (hereinafter "Guizzo article"). The most relevant passages are believed to be on pages 4 and 5 of the attached print-out and are set forth below:

... [T]urbo codes use two component decoders that work together to bypass the complexity problem.

The role of each decoder is to get the data, which might have been corrupted by noise along the channel, and decide which is the more likely value, 0 or 1, for each individual bit. ...

...

...[I]t turns out that the reliability information of one decoder is useful to the other and vice versa, because the two strings of parity bits refer to the very same data; it's just that the bits are arranged in a different order. So the two decoders are trying to solve the same problem but looking at it from different perspectives.

The two decoders, then, can exchange reliability information in an iterative way to improve their own decoding. All they have to do, before swapping reliability strings, is arrange the strings' content in the order each decoder needs. So a bit that was strongly detected as a 1 in one decoder, for example, influences the other decoder's opinion on the corresponding bit.

...

At the heart of turbo coding is this iterative process, in which each component decoder takes advantage of the work of the other at a previous decoding step. After a certain number of iterations, typically four to 10, both decoders begin to agree on all bits. ...

Thus, the Guizzo article confirms that an essential part of turbo decoding is the use of two separate, cooperating decoders.

Applicants respectfully request that the Examiner carefully consider the articles appended hereto, as well as the additional passage in the Heegard book cited by the applicants hereinabove. If the Examiner does so, it is believed that he will accept the applicants' contention that turbo decoding requires at least two separate, cooperating decoders. Since no such decoding apparatus is disclosed or discussed in the Scheuermann reference, it follows a fortiori that Scheuermann fails to disclose an instruction decoder operative to control an ACS engine to perform turbo decoding. This point, by itself, is believed to be sufficient to require that the pending rejections be withdrawn.

However, applicants believe that the Scheuermann reference is inadequate to support the Examiner's reliance thereon in at least one other respect. The Examiner's attention is again respectfully directed to the language of claim 1, which specifies an ACS engine that performs one type of decoding (Viterbi decoding) in response to an instruction decoder receiving a first instruction, and that performs a second type of decoding (turbo decoding) in response to the instruction decoder receiving a second instruction. Thus, putting aside for the moment that Scheuermann does not concern itself with turbo decoding, Scheuermann is further deficient in that it does not disclose two different types of decoding, nor selectively operating an ACS engine to perform two different types of decoding. In considering the applicants' arguments, the Examiner is respectfully requested to focus not only on the lack of turbo decoding in Scheuermann's disclosure but also on Scheuermann's failure to disclose two types of decoding and selecting between two types of decoding. Further, nothing in the reference teaches or

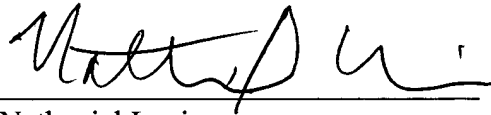
suggests using the same ACS engine to perform two different types of decoding. In short, there are reasons for withdrawing the rejections beyond the lack of any discussion in the Scheuermann reference of turbo decoding.

Reconsideration and withdrawal of the pending rejections are respectfully requested.

CONCLUSION

Accordingly, Applicants respectfully request allowance of the pending claims. If any issues remain, or if the Examiner has any further suggestions for expediting allowance of the present application, the Examiner is kindly invited to contact the undersigned via telephone at (203) 972-3460.

Respectfully submitted,



Nathaniel Levin
Registration No. 34,860
Buckley, Maschoff & Talwalkar LLC
Attorneys for Intel Corporation
Five Elm Street
New Canaan, CT 06840
(203) 972-3460

July 11, 2006
Date



EXHIBIT A

Turbo Code

Turbo code

From Wikipedia, the free encyclopedia

Turbo codes are a class of recently-developed high-performance error correction codes finding use in deep-space satellite communications and other applications where designers seek to achieve maximal information transfer over a limited-bandwidth communication link in the presence of data-corrupting noise.

Contents

- 1 The Shannon limit
- 2 Who devised Turbo Codes?
- 3 How turbo codes work
- 4 The encoder
- 5 The decoder
- 6 Solving hypotheses to find bits
- 7 Practical applications using Turbo Codes
- 8 Bayesian formulation
- 9 External links

The Shannon limit

Of all practical error correction methods known to date, turbo codes, together with Low-density parity-check codes, come closest to approaching the Shannon limit, the theoretical limit of maximum information transfer rate over a noisy channel.

Turbo codes make it possible to increase data rate without increasing the power of a transmission, or they can be used to decrease the amount of power used to transmit at a certain data rate. Its main drawbacks are the relative high decoding complexity and a relatively high latency, which makes it unsuitable for some applications.

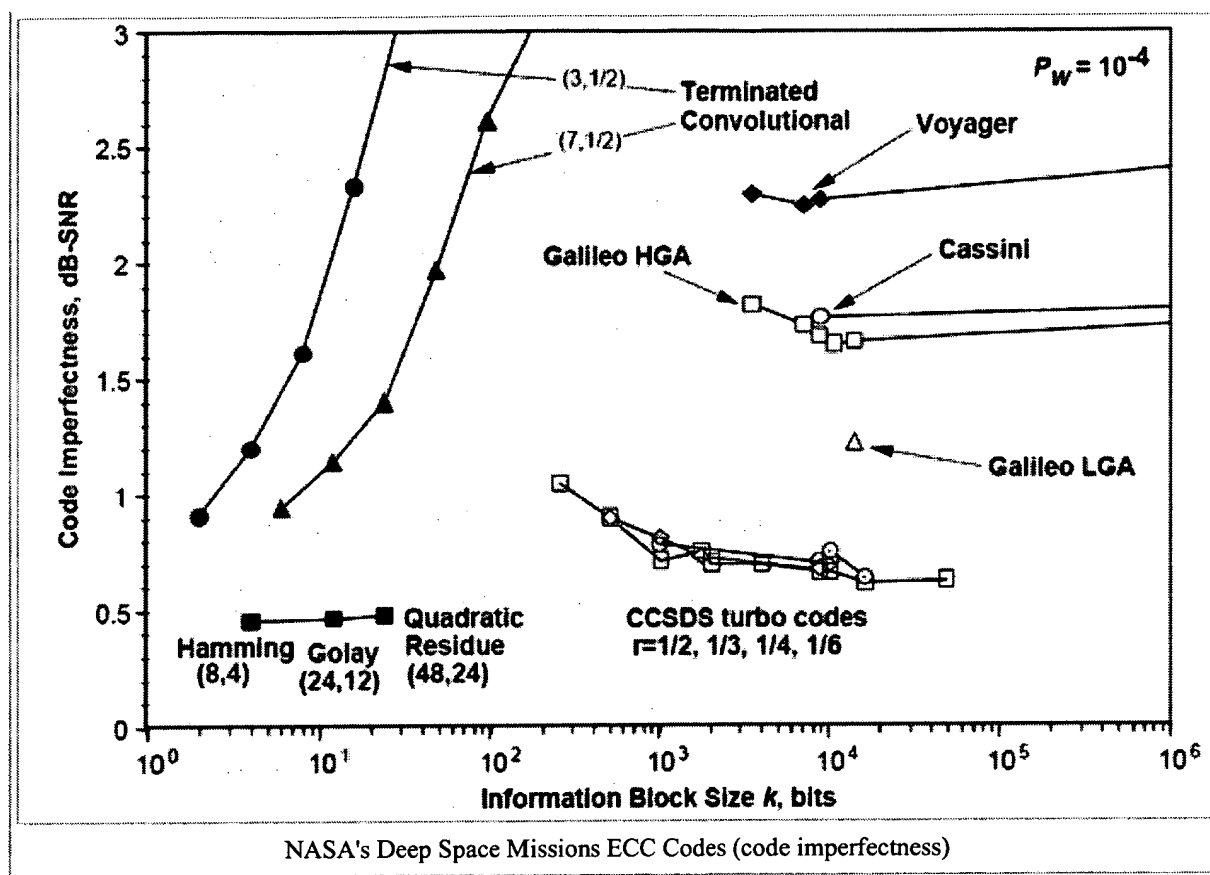
- For satellite use, this is not of great concern, since the transmission distance itself introduces latency due to the limited speed of light.

Prior to Turbo codes, the best known technique combined a Reed-Solomon error correction block code with a Viterbi algorithm convolutional code, also known as RSV codes. These RSV codes never were able to approach the Shannon limit as closely as Turbo codes have been able to.

Who devised Turbo Codes?

The method was introduced by Berrou, Glavieux, and Thitimajshima (from ENST Bretagne, France) in their 1993 paper: "*Near Shannon Limit error-correcting coding and decoding: Turbo-codes*" published in the Proceedings of IEEE International Communications Conference [1] (<http://www-elec.enst-bretagne.fr/equipe/berrou/Near%20Shannon%20Limit%20Error.pdf>). Turbo code refinements and implementation are an area of active research at a number of universities.

How turbo codes work



There are two related features of turbo codes that make them different from the more traditional error-correcting codes of the 20th century

- The key insight is the realization that instead of producing a stream of binary digits from the signal it receives, the front-end of the decoder can be designed to produce a likelihood measure for each bit.
- The nitty-gritty of turbo codes is the design of the decoder (and the coder) so that it can exploit this additional information.

The encoder

The encoder sends three sub-blocks of bits. The first sub-block is the m -bit block of payload data. The second sub-block is $n/2$ parity bits for the payload data, computed using a convolutional code. The third sub-block is $n/2$ parity bits for a known permutation of the payload data, again computed using a convolutional code. That is, two redundant but different sub-blocks of parity bits for the sent payload. The complete block has $m+n$ bits of data with a code rate of $m/(m+n)$.

The decoder

The decoder front-end produces an integer for each bit in the data stream. This integer is a measure of how likely it is that the bit is a 0 or 1 and is also called *soft bit*. The integer could be drawn from the range $[-127, 127]$, where:

- -127 means "certainly 0"

- -100 means "very likely 0"
- 0 means "it could be either 0 or 1"
- 100 means "very likely 1"
- 127 means "certainly 1"
- etc

This introduces a probabilistic aspect to the data-stream from the front end, but it conveys more information about each bit than just 0 or 1.

For example, for each bit, the front end of a traditional wireless-receiver has to decide if an internal analog voltage is above or below a given threshold voltage level. For a turbo-code decoder, the front end would provide an integer measure of how far the internal voltage is from the given threshold.

To decode the $m+n$ -bit block of data, the decoder front-end creates a block of likelihood measures, with one likelihood measure for each bit in the data stream. There are two parallel decoders, one for each of the $n/2$ -bit parity sub-blocks. Both decoders use the sub-block of m likelihoods for the payload data. The decoder working on the second parity sub-block knows the permutation that the coder used for this sub-block.

Solving hypotheses to find bits

The nitty gritty of turbo codes is how they use the likelihood data to reconcile differences between the two decoders. Each of the two convolutional decoders generates a hypothesis (with derived likelihoods) for the pattern of m bits in the payload sub-block. The hypothesis bit-patterns are compared, and if they differ, the decoders exchange the derived likelihoods they have for each bit in the hypotheses. Each decoder incorporates the derived likelihood estimates from the other decoder to generate a new hypothesis for the bits in the payload. Then they compare these new hypotheses. This iterative process continues until the two decoders come up with the same hypothesis for the m -bit pattern of the payload, typically in 15 to 18 cycles.

An analogy can be drawn between this process and that of solving cross-reference puzzles like crossword or sudoku. Consider a partially-completed, possibly garbled crossword puzzle. Two puzzle solvers (decoders) are trying to solve it: one possessing only the "down" clues (parity bits), and the other possessing only the "across" clues. To start, both solvers guess the answers (hypotheses) to their own clues, noting down how confident they are in each letter (payload bit). Then, they compare notes, by exchanging answers and confidence ratings with each other, noticing where and how they differ. Based on this new knowledge, they both come up with updated answers and confidence ratings, repeating the whole process until they converge to the same solution.

Practical applications using Turbo Codes

Telecommunications:

- Turbo codes are used extensively in 3G mobile telephony standards.
- Future DVB-S (Satellite television) standards will use Turbo Codes to replace the Reed-Solomon (RS) codes now used.
- Some future NASA missions will use Turbo Codes as standard, replacing concatenated RS-Viterbi codes.

Bayesian formulation

From an artificial intelligence viewpoint, turbo codes can be considered as an instance of loopy belief propagation in bayesian networks.

External links

- Turbo code homepage (<http://www331.jpl.nasa.gov/public/>) at the JPL
- "The UMTS Turbo Code and an Efficient Decoder Implementation Suitable for Software-Defined Radios" (<http://www.csee.wvu.edu/~mvalenti/documents/valenti01.pdf>) (*International Journal of Wireless Information Networks*)
- Coded Modulation Library (<http://www.iterativesolutions.com/Matlab.htm>), an open source library for simulating turbo codes in matlab
- "Pushing the Limit" (<http://www.sciencenews.org/articles/20051105/bob8.asp>), a *Science News* feature about the development and genesis of turbo codes
- "Closing in on the perfect code" (<http://web.archive.org/web/http://www.spectrum.ieee.org/WEBONLY/publicfeature/mar04/0304cod> at the Wayback Machine (originally published in *IEEE Spectrum*)
- Dana Mackenzie (2005). "Take it to the limit". *New Scientist* **187** (2507): 38–41. ISSN 0262-4079 (<http://dispatch.opac.ddb.de/DB=1.1/LNG=EN/CMD?ACT=SRCHA&IKT=8&TRM=0262-4079>). (preview (<http://www.newscientist.com/article.ns?id=mg18725071.400>), copy (http://geilenkotten.homeunix.org/TC_NS_09072005.pdf))

Retrieved from "http://en.wikipedia.org/wiki/Turbo_code"

Category: Error detection and correction

- This page was last modified 17:08, 23 June 2006.
- All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc.



EXHIBIT B

Pushing the Limit: Digital communications experts are zeroing in on the perfect code (by Erica Klarreich)

[Home](#) [Table of Contents](#) [Feedback](#) [Subscribe](#) [Help/About](#) [Archives](#) [Search](#)

SCIENCE NEWS Online

THE WEEKLY NEWSMAGAZINE OF SCIENCE

SUBSCRIBER SERVICES**SUBSCRIBE****RENEW****Change of Address****Classroom Subscriptions****Contact Us****Gift Subscriptions****Order Back Issues****WEB FEATURES****Archives****Audio (Podcasting)****Blog****Book Listings****E-mail Alert****Most Viewed Articles****Science News of the Year****ABOUT SCIENCE NEWS****Advertise****Copyright Permissions****History****Merchandise**☐ **Print Article**☐ **E-mail Article**

Week of Nov. 5, 2005; Vol. 168, No. 19, p. 29

Pushing the Limit

Digital communications experts are zeroing in on the perfect code

Erica Klarreich

When SMART-1, the European Space Agency's first mission to the moon, launched in September 2003, astronomers hailed it as the testing ground for a revolutionary and efficient solar-electric-propulsion technology. While this technological leap absorbed the attention of scientists and the news media, a second, quieter revolution aboard SMART-1 went almost unheralded. Only a small band of mathematicians and engineers appreciated the quantum leap forward for digital communications. Incorporated into SMART-1's computers was a system for encoding data transmissions that, in a precise mathematical sense, is practically perfect.



Science News
for Kids

audible.com®

Subscribe to an
audio format

SUBSCRIBE!



**So many
great
topics...**

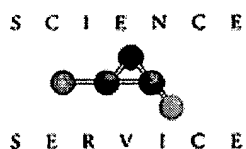
**... in as little as
15 minutes a week**



SMART TALKER. The European Space Agency's SMART-1 mission to the moon, depicted in this drawing, is testing a solar-electric-propulsion technology. The bigger news for mathematicians is that the craft is sending those data to Earth via a digital code that's exceptionally efficient.
European Space Agency

Published by

Space engineers have long grappled with the problem of how to reliably transmit data, such as pictures and scientific measurements, from space probes back to Earth. How can messages travel hundreds of millions of



miles without the data becoming hopelessly garbled by noise? In a less extreme situation, as any cell phone user can attest, noise is also an issue for communication systems on Earth.

Over the past half-century, mathematicians and computer scientists have come up with clever approaches to the noise problem. They've devised codes that intentionally incorporate redundancy into a message so that even if noise corrupts some portions, the recipient can usually figure it out. Called error-correcting codes, these underlie a host of systems for digital communication and data storage, including cell phones, the Internet, and compact disks. Space missions use this technique to send messages from transmitters that are often no more powerful than a dim light bulb.

Yet coding theorists have been aware that their codes fall far short of what can, in theory, be achieved. In 1948, mathematician Claude Shannon, then at Bell Telephone Laboratories in Murray Hill, N.J., published a landmark paper in which he set a specific goal for coding theorists. Shannon showed that at any given noise level, there is an upper limit on the ratio of the information to the redundancy required for accurate transmission. As with the speed of light in physics, this limit is unattainable, but—in theory at least—highly efficient codes can come arbitrarily close.

The trouble was that no one could figure out how to construct the superefficient codes that Shannon's theory had predicted. By the early 1990s, state-of-the-art codes were typically getting information across at only about half the rate that Shannon's law said was possible.

Then, in the mid-1990s, an earthquake shook the coding-theory landscape. A pair of French engineers—outsiders to the world of coding theory—astonished the insiders with their invention of what they called turbo codes, which come within a hair's breadth of Shannon's limit.

Later in the decade, coding theorists realized that a long-forgotten method called low-density parity-check (LDPC) coding could edge even closer to the limit.

Turbo codes and LDPC codes are now coming into play. In addition to being aboard the SMART-1 mission, turbo-code technology is on its way to Mercury on NASA's Messenger mission, launched last year. In the past couple of years, turbo codes have also found their way into millions of mobile phones, enabling users to send audio and video clips and to surf the Internet.

LDPC codes, meanwhile, have become the new standard for digital-satellite television. Hundreds of research groups are studying potential applications of the two kinds of codes at universities and industry giants including Sony, Motorola, Qualcomm, and Samsung.

"In the lab, we're there" at the Shannon limit, says Robert McEliece, a coding theorist at the California Institute of Technology in Pasadena. "Slowly, the word is getting out, and the technology is being perfected."

The closing of the gap between state-of-the-art codes and the Shannon limit could save space agencies tens of millions of dollars on every mission because they could use lighter data transmitters with smaller batteries and antennas. For cell phone and other wireless technologies, the new codes could reduce the number of dead spots in the world and lower the cost of service.

Noisy communication

Most engineering fields start with inventions; researchers then gradually figure out the limits of what can be achieved. Coding theory experienced the reverse. The field got launched by Shannon's result, which used theoretical insights to place an absolute limit on what the clever coders of the future could achieve.

Shannon considered how much redundancy must be added to a message so that the information in it can survive a noisy transmission. One way to fight noise is simply to send the same message several times. For example, if a friend can't hear what you're saying at a noisy party, you might repeat yourself. However, it might be more effective to rephrase your message. Coding theorists look for effective ways to rephrase the information in a message to get it across using the smallest number of bits—zeros and ones.

For example, to protect a message against a single transmission error, one solution is to send the message three times. However, that requires three times as many bits as the original did. In 1950, Richard Hamming of Bell Laboratories came up with a much more efficient approach that adds only three redundant bits for every four bits of the original message.

In Hamming's approach, each of the three added bits says something about a particular combination of bits in the original message. Given a four-bit message, bit number 5 is chosen in such a way that bits 1, 2, 3, and 5 will have an even number of ones. Bit 6 is chosen to make bits 1, 2, 4, and 6 have an even number of ones, and bit 7 is chosen to make bits 2, 3, 4, and 7 have an even number of ones. Chase through the possibilities, and you'll find that if noise flips one of the seven transmitter bits from 0 to 1, or vice versa, even-odd checks reveal which bit is wrong.

Over the decades, coding theorists have come up with even more-efficient schemes and ones that can cope with a higher error rate. Shannon's law, however, says that there is a limit to how good these codes can get—whether the communication channel is a fiber-optic cable or a noisy room. Try to send more information and less

redundancy than the channel can support, and the message won't survive the channel's noise.

"Shannon showed that even with infinite computing power, infinitely bright engineers, and no constraints on your budget, there is this absolute limit on what you can achieve," McEliece says.

Shannon's law has a bright side, however. He proved that almost every code whose efficiency rate is below the Shannon limit would permit the recipient to recover the original message almost perfectly.

But there's a catch. Shannon didn't provide a practical recipe for constructing reliable codes that come close to the Shannon limit. In the decades that followed, researchers tried but failed to develop such codes. Coding theorists became fond of saying that almost all codes are good—except the ones they know about.

Turbo boost

In 1993 at the IEEE International Conference on Communications in Geneva, a pair of French engineers made an incredible claim. They reported that they had come up with a coding method that could provide almost perfect reliability at a rate breathtakingly close to the Shannon limit. These "turbo" codes, the researchers asserted, could transmit data about twice as fast as other codes do or, alternatively, could use only half as much transmitting power to achieve the same rates as other codes do.

"Their simulation curves claimed unbelievable performance, way beyond what was thought possible," McEliece says.

Many experts at the conference scoffed at the claims and didn't bother attending the engineers' talk. "They said we must have made an error in our simulations," recalls Claude Berrou of the French National School of Telecommunications in Brest, who invented turbo codes together with his colleague the late Alain Glavieux.

Within a year, however, coding theorists were reproducing the French engineers' results. "Suddenly, the whole world of coding theory started twinkling and scintillating with the news," says Michael Tanner, a coding theorist at the University of Illinois at Chicago. "Turbo codes changed the whole problem."

"The thing that blew everyone away about turbo codes is not just that they get so close to Shannon capacity but that they're so easy. How could we have overlooked them?" McEliece says. "I think it was a case of fools rushing in where angels fear to tread. Berrou and Glavieux didn't know the problem was supposed to be hard, so they managed to find a new way to go about it."

Turbo codes use a divide-and-conquer approach in which each of two decoders gets a different encoded version of the original message and the decoders then collaborate. Berrou likens the code to a crossword puzzle in which one would-be solver receives the "across" clues and another receives the "down" clues.

In the first round, each decoder uses its own clues to guess the bits of the original message, keeping track of how confident it is about each guess. Next, the two decoders compare notes and each updates its guesses and confidence levels. In the analogy with crosswords, if you guessed that an across word was *wolverine*, and a friend told you that a down clue also put a *w* in the first square, your confidence in your guess would grow. However, if you learned that the down clue put, say, a *p* in the first square, you would feel less confident about *wolverine* or possibly switch your guess to another word.

After the two decoders update their proposed solutions, they compare notes again and then update their solutions again. They continue this process until they reach a consensus about the original message. This typically takes 4 to 10 passes of information back and forth.

Rather than give two decoders partial information, it might seem more efficient to send all the clues to a single decoder, in closer analogy with how crosswords are usually solved. In principle, this would result in more-accurate decoding of the message than the two turbo decoders could produce. In practice, however, as the number of clues increases, the number of computations needed in the decoding process grows exponentially, so sending all the clues to one decoder would result in too slow a decoding process.

In hindsight, McEliece says, coding theory before 1993 was overly focused on the optimal, but inefficient, decoding that a single decoder offers.

"The genius of turbo codes is that Berrou and Glavieux said, 'Let's not talk about the best possible decision but about a good-enough decision,'" McEliece says. "That's a much less complex problem, and they got it to work."

Parity regained

Although turbo codes quickly became established as the fastest codes around, they soon had to share that title. Within a few years, coding theorists realized that a slightly modified version of an old, obscure code could sometimes produce even better results.

In 1958, Robert Gallager, then a Ph.D. student at the Massachusetts Institute of Technology (MIT), created a class of codes that, like Hamming's codes, add redundant

bits that permit decoders to do even-odd checks to guess which bits of the original message have been corrupted by noise. As with turbo codes, Gallager's LDPC codes use cross talk between decoders to gradually establish confidence about the likely bits of the original message. However, in contrast to turbo codes, which use just two decoders, LDPC codes use a separate decoder for each bit of the message, creating a giant rumor mill. That requires thousands, or even tens of thousands, of decoders.

"It's like turbo codes balkanized to the nth degree," McEliece says.

Gallager's simulations showed that LDPC codes come very close to the Shannon limit. However, his method was impractical at the time because it presented a computational problem far too complex for the computers of the day.

"The very biggest computer at MIT in the early sixties had an amount of computer power only about equivalent to what you get in a cell phone today," recalls Gallager, now an MIT professor. Because of this limitation, most coding theorists regarded LDPC codes as intriguing but quixotic—and promptly forgot about them.

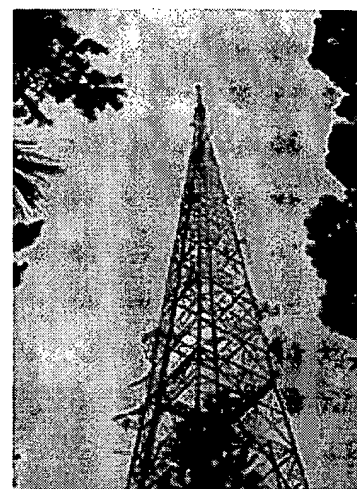
LDPC codes are "a piece of 21st-century coding that happened to fall in the 20th century," says David Forney, a coding theorist at MIT. In the 1960s, he worked at Codex Corp. in Newton, Mass., a Motorola subsidiary that held the patents to LDPC codes. "I'm embarrassed to say we never once seriously considered using those patents," he admits.

In the mid-1990s, however, with the coding-theory world buzzing about turbo codes' cross talk approach, explorations of several research groups independently led them back to Gallager's work.

"Gallager's paper was almost clairvoyant," McEliece says. "Every important concept about LDPC codes is buried in there, except one."

That one concept is irregularity, in which there are more even-odd checks on certain bits of the original message than on others. For a crossword puzzle, this would be like having a particular letter of the grid be defined not just by across and down clues but also by a diagonal clue.

The extra clues of irregular LDPC codes permit the decoders to guess the value of certain bits with an extremely high confidence level. The



CODING POWER. Thanks to digital-coding advances, cell-phone towers such as this one in Arlington, Va., are carrying audio and video clips and information from the Internet.

S. Norcross

decoders can then use these high-confidence bits to become more confident about the other bits of the message, creating a "ripple effect," says Amin Shokrollahi of the Federal Polytechnic School of Lausanne in Switzerland, one of the coding theorists who pioneered the concept of irregularity. With this innovation, LDPC codes sometimes edge out turbo codes in the race toward the Shannon limit.

The ideas behind turbo codes and LDPC codes have rendered much of the preceding 50 years of coding theory obsolete, says David MacKay of Cambridge University in England, one of the coding theorists who rediscovered LDPC codes. "Future generations won't have to learn any of the stuff that has been the standard in textbooks," he says.

To some coding theorists, turbo codes and LDPC codes represent the end of coding theory. In the lab, they've reached within 5 percent of the maximum efficiency, and commercial systems are within 10 percent, says McEliece.

"We're close enough to the Shannon limit that from now on, the improvements will only be incremental," says Thomas Richardson, a coding theorist at Flarion Technologies in Bedminster, N.J. "This was probably the last big leap in coding."

In 1,000 years, McEliece whimsically suggests, the *Encyclopedia Galactica* will sweep past most of the past half-century of coding theory to say simply, "Claude Shannon formulated the notion of channel capacity in 1948 A.D. Within several decades, mathematicians and engineers had devised practical ways to communicate reliably at data rates within 1 percent of the Shannon limit...."

If you have a comment on this article that you would like considered for publication in *Science News*, send it to editors@sciencenews.org. Please include your name and location.

To subscribe to *Science News* (print), go to
<https://www.kable.com/pub/scnw/subServices.asp>.

To sign up for the free weekly e-LETTER from *Science News*, go to
http://www.sciencenews.org/pages/subscribe_form.asp.



References:

Berrou, C., A. Glavieux and P. Thitimajshima. 1993. Near Shannon limit error-correcting coding: Turbo codes. Proceedings of the IEEE International Conference on Communications. May. Geneva. Reprint

available at <http://www-elec.enst-bretagne.fr/equipe/berrou/Near%20Shannon%20Limit%20Error.pdf>.

Gallager, R.G. 1963. *Low Density Parity Check Codes*. Cambridge, Mass.: MIT Press.

Luby, M.G., M. Mitzenmacher, M.A. Shorollahi, and D.A. Spielman. 1998. Analysis of low-density codes and improved designs using irregular graphs. 30th ACM Symposium on the Theory of Computing Dallas. Available at <http://www.cs.yale.edu/homes/spielman/Research/irreg.html>.

MacKay, D.J.C., and R. M. Neal. 1996. Near Shannon limit performance of low density parity check codes. *Electronic Letters* 32 (Aug. 29): 1645-1646. Abstract available at <http://dx.doi.org/10.1049/el:19961141>.

Further Readings:

Klarreich, E. 2004. Oddballs. *Science News* 166(Oct. 2):219-221. Available to subscribers at <http://www.sciencenews.org/articles/20041002/bob9.asp>.

Peterson, I. 2003. SET math. *Science News Online* (Aug. 23). Available at <http://www.sciencenews.org/articles/20030823/mathtrek.asp>.

_____. 2002. DNA's error-detecting code. *Science News Online* (Sept. 21). Available at <http://www.sciencenews.org/articles/20020921/mathtrek.asp>.

_____. 1997. Deep in the jungles. *Science News Online* (Sept. 13). Available at http://www.sciencenews.org/pages/sn_arc97/9_13_97/mathland.htm

Sources:

Claude Berrou
Electronics Department
French National School of Telecommunications
Brest
France

G. David Forney Jr.
Department of Electrical Engineering and Computer Science
Laboratory for Information and Information Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Robert G. Gallager

Department of Electrical Engineering and Computer Science
Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
BLD 32 - D628
Cambridge, MA 02139

David J.C. MacKay
Department of Physics
Cavendish Laboratory
University of Cambridge
Madingley Road
Cambridge CB3 0HE
United Kingdom

Robert J. McEliece
Department of Electrical Engineering (136-93)
California Institute of Technology
Pasadena, CA 91125

Thomas Richardson
Flarion Technologies
Bedminster One
135 Route 202/206 South
Bedminster, NJ 07921

M. Amin Shokrollahi
Digital Fountain, Inc.
39141 Civic Center Drive
Fremont, CA 94538

R. Michael Tanner
Academic Affairs
University of Illinois, Chicago
2832 University Hall (MC-105)
601 South Morgan
Chicago, IL 60607

From *Science News*, Vol. 168, No. 19, Nov. 5, 2005, p. 296.

[Home](#) | [Table of Contents](#) | [Feedback](#) | [Subscribe](#) | [Help/About](#) | [Archives](#) | [Search](#)

For customer service and subscription orders please call 1-800-552-4412

Copyright ©2005 Science Service. All rights reserved.

1719 N St., NW, Washington, DC 20036 | 202-785-2255 | scinews@sciserv.org



EXHIBIT C

Closing in on the perfect code (by Erico Guizzo)

Best Available Copy

IEEE HOME | SEARCH IEEE | SHOP | WEB ACCOUNT | CONTACT IEEE

Membership

Publications

Standards

Careers/Jobs

IEEE
SPECTRUM ONLINE[Member Table of Contents](#) [Guest Table of Contents](#) [Job Site](#) [Editorial Staff](#) [Advertising](#) [Direct](#)

Wed 05 Jul 06 17:25 -400 GMT

[Home](#) >> [Feature Story](#)[FEATURE ARTICLE](#)

SITE IN

WEEKLY FEATURE

Closing in on the perfect code

Turbo codes, which let engineers pump far more error-free data through a channel, will be the key to the next generation of multimedia cellphones

By Erico Guizzo

It's not often in the rarefied world of technological research that an esoteric paper is greeted with scoffing. It's even rarer that the paper proves in the end to be truly revolutionary.

It happened a decade ago at the 1993 IEEE International Conference on Communications in Geneva, Switzerland. Two French electrical engineers, Claude Berrou and Alain Glavieux, made a flabbergasting claim: they had invented a digital coding scheme that could provide virtually error-free communications at data rates and transmitting-power efficiencies well beyond what most experts thought possible.

The scheme, the authors claimed, could double data throughput for a given transmitting power or, alternatively, achieve a specified communications data rate with half the transmitting energy—a tremendous gain that would be worth a fortune to communications companies.

Few veteran communications engineers believed the results. The Frenchmen, both professors in the electronics department at the Ecole Nationale Supérieure des Télécommunications de Bretagne in Brest, France, were then unknown in the information-theory community. They must have gone astray in their calculations, some reasoned. The claims were so preposterous that many experts didn't even bother to read the paper.

Unbelievable as it seemed, it soon proved true, as other researchers began to replicate the results. Coding experts then realized the significance of that work. Berrou and Glavieux were right, and their error-correction coding scheme, which has since been dubbed turbo codes, has revolutionized error-correction coding. Chances are fairly good that the next cellphone you buy will have them built in.

From a niche technology first applied mainly in satellite links and in at least one deep-space communications system, turbo codes are about to go mainstream. As they are incorporated

into the next-generation mobile telephone system, millions of people will soon have them literally in their hands. This coding scheme will let cellphones and other portable devices handle multimedia data such as video and graphics-rich imagery over the noisy channels typical of cellular communications. And researchers are studying the use of turbo codes for digital audio and video broadcasting, as well as for increasing data speeds in enhanced versions of Wi-Fi networks.

With possibilities like these, turbo codes have jumped to the forefront of communications research, with hundreds of groups working on them in companies and universities all over the world. The list includes telecommunications giants like France Télécom and NTT DoCoMo; high-tech heavyweights like Sony, NEC, Lucent, Samsung, Ericsson, Nokia, Motorola, and Qualcomm; hardware and chip manufacturers like Broadcom, Conexant, Comtech AHA, and STMicroelectronics; and start-ups like Turboconcept and iCoding.

Turbo codes do a simple but incredible thing: they let engineers design systems that come extremely close to the so-called channel capacity—the absolute maximum capacity, in bits per second, of a communications channel for a given power level at the transmitter. This threshold for reliable communications was discovered by the famed Claude Shannon, the brilliant electrical engineer and mathematician who worked at Bell Telephone Laboratories in Murray Hill, N.J., and is renowned as the father of information theory [see sidebar, "[Shannon: Cracking the Channel](#)"].

In a landmark 1948 paper, Shannon, who died in 2001, showed that with the right error-correction codes, data could be transmitted at speeds up to the channel capacity, virtually free from errors, and with surprisingly low transmitting power. Before Shannon's work, engineers thought that to reduce communications errors, it was necessary to increase transmission power or to send the same message repeatedly—much as when, in a crowded pub, you have to shout for a beer several times.

Shannon basically showed it wasn't necessary to waste so much energy and time if you had the right coding schemes. After his discovery, the field of coding theory thrived, and researchers developed fairly good codes. But still, before turbo codes, even the best codes usually required more than twice the transmitting power that Shannon's law said was necessary to reach a certain level of reliability—a huge waste of energy. The gap between the practical and the ideal, measured in decibels—a ratio between the signal level and the noise level on a logarithmic scale—was about 3.5 dB. To chip away at it, engineers needed more elaborate codes.

That was the goal that persisted for more than four decades, until Berrou and Glavieux made their discovery in the early 1990s. When they introduced turbo codes in 1993, they showed it was possible to get within an astonishing 0.5 dB of the Shannon limit, for a bit-error rate of one in 100 000. Today, turbo codes are still chipping away at even that small gap.

The solution to overcoming the noise that plagued all communications channels, according to Shannon's seminal paper, was to divide the data into strings of bits and add to each string a set of extra bits—called parity bits—that would help identify and correct errors at the receiving end. The resulting group of bits—the data bits plus the parity bits—is called a codeword, and typically it represents a block of characters, a few image pixels, a sample of voice, or some other piece of data.

Shannon showed that with the right collection of codewords—with the right code, in other words—it was possible to attain the channel capacity. But then, which code could do it? "Shannon left unanswered the question of inventing codes," says David Forney, a professor of electrical engineering at the Cambridge-based Massachusetts Institute of Technology (MIT) and an IEEE Fellow. Shannon proved mathematically that coding was the means to reach capacity, but he didn't show exactly how to construct these capacity-approaching codes. His

work, nevertheless, contained valuable clues.

Shannon thought of codewords as points in space. For example, the codeword 011 can be considered a point in a three-dimensional space with coordinates $x = 0$, $y = 1$, and $z = 1$. Codewords with more than three bits are points in hyperspace. Noise can alter a codeword's bits, and therefore its coordinates, displacing the point in space. If two points are close to each other and one is affected by noise, this point might fall exactly onto the other, resulting in decoding error. Therefore, the larger the differences in codewords—the farther apart they are—the more difficult it is for noise to cause errors.

To achieve capacity, Shannon demonstrated that you should randomly choose infinitely long codewords. In other words, going back to his spatial analogy, if you could make the codewords both random and as long as you wanted, you could put the points arbitrarily far from each other in space. There would be essentially no possibility of one point erroneously falling on another. Unfortunately, such long, random codes are not practical: first, because there is an astronomical number of codewords; second, because this code would be extremely slow to use as you transmitted many, many bits for just one codeword. Still, the random nature of a good code would turn out to be critical for turbo codes.

Coding experts put aside Shannon's ideal random codes, as they concentrated on developing practical codes that could be implemented in real systems. They soon began to develop good codes by cleverly choosing parity bits that constrained codewords to certain values, making these codewords unlikely to be confused with other ones.

For example, suppose we have an eight-bit codeword (seven data bits plus one parity bit). Suppose we further insist that all the codewords have an even number of 1s, making that extra parity bit a 1 if necessary to fulfill that requirement. Now, if any of the eight bits is altered by noise, including the parity bit itself, the receiver knows there was an error, because the parity count won't check—there would be an odd number of 1s.

This basic scheme can detect an error, but it can't correct it—you don't know which bit was flipped. To correct errors, you need more parity bits. Coding experts have come up with numerous and ever more sophisticated ways of generating parity bits. Block codes, Hamming codes, Reed-Solomon codes, and convolutional codes are widely used and achieve very low error rates.

Nevertheless, a computational-complexity problem hounded coding specialists and plagued all these codes. The complexity problem emerges as you figure the cost of a code in terms of the amount of computation required to decode your data. The closer you get to Shannon's limit, the more complicated this process becomes, because you need more parity bits and the codewords get longer and longer.

For codewords with just 3 bits, for instance, you have a total of only 2^3 , or 8, codewords. To approach capacity, however, you might need codewords with, say, 1000 bits, and therefore your decoder would need to search through an unimaginably large collection of 2^{1000} —approximately 10^{301} —codewords. For comparison, the estimated number of atoms in the visible universe is about 10^{80} .

The upshot was that if you set about exploiting the best existing codes as your strategy for achieving arbitrarily reliable communications at Shannon's limit, you would be doomed to failure. "The computational complexity is just astronomical," says IEEE Fellow R. Michael Tanner, a professor of electrical and computer engineering and provost at the University of Illinois at Chicago. "These codes don't have the capability to do it." How could researchers get past this barrier? It was hopeless, some actually concluded in the late 1970s.

Turbo codes solved the complexity problem by splitting it into more manageable

components. Instead of a single encoder at the transmitter and a single decoder at the receiver, turbo codes use two encoders at one end and two decoders at the other [see illustration, "[How Turbo Codes Work](#)"].

Researchers had realized in the late 1960s that passing data through two encoders in series could improve the error-resistance capability of a transmission—for such a combination of encoders, the whole is more than the sum of the parts. Turbo codes employ two encoders working synergistically—not in series, but in parallel.

The turbo process starts with three copies of the data block to be transmitted. The first copy goes into one of the encoders, where a convolutional code takes the data bits and computes parity bits from them. The second copy goes to the second encoder, which contains an identical convolutional code. This second encoder gets not the original string of bits but rather a string with the bits in another order, scrambled by a system called an interleaver. This encoder then reads these scrambled data bits and computes parity bits from them. Finally, the transmitter takes the third copy of the original data and sends it, along with the two strings of parity bits, over the channel.

That rearranging of the bits in the interleaver is the key step in the whole process. Basically, this permutation brings more diversity to the codewords; in the spatial analogy, it pushes the points farther apart in space. "The role of the permutation is to introduce some random behavior in the code," says Berrou. In other words, the interleaver adds a random character to the transmitted information, much as Shannon's random codes would do.

But then turbo codes, like any other code with a huge number of codewords, would also hit the wall of computational complexity. In fact, turbo codes usually work with codewords having around a thousand bits, a fairly unwieldy number. Hopeless? Yes, if you had a single decoder at the receiver. But turbo codes use two component decoders that work together to bypass the complexity problem.

The role of each decoder is to get the data, which might have been corrupted by noise along the channel, and decide which is the more likely value, 0 or 1, for each individual bit. In a sense, deciding about the value of each bit is as if you had to guess whether it's raining or not outside. Suppose you can't look out a window and you don't hear any sounds; in this case, you basically have no clue, and you can simply flip a coin and make your guess. But what if you check the forecast and it calls for rain? Also, what if you suddenly hear thunder? These events affect your guess. Now you can do better than merely flipping a coin; you'll probably say there's a good chance that it is raining and you will take your umbrella with you.

Each turbo decoder also counts on "clues" that help it guess whether a received bit is a 0 or a 1. First, it inspects the analog signal level of the received bits. While many decoding schemes transform the received signal into either a 0 or a 1—therefore throwing away valuable information, because the analog signal has fluctuations that can tell us more about each bit—a turbo decoder transforms the signal into integers that measure how confident we can be that a bit is a 0 or a 1. In addition, the decoder looks at its parity bits, which tell it whether the received data seems intact or has errors.

The result of this analysis is essentially an informed guess for each bit. "What turbo codes do internally is to come up with bit decisions along with reliabilities that the bit decisions are correct," says David Garrett, a researcher in the wireless research laboratory at Bell Labs, part of Lucent Technologies, Murray Hill, N.J. These bit reliabilities are expressed as numbers, called log-likelihood ratios, that can vary, for instance, between -7 and +7. A ratio of +7 means the decoder is almost completely sure the bit is a 1; a -5 means the decoder thinks the bit is a 0 but is not totally convinced. (Real systems usually have larger intervals, like -127 to +127.)

Even though the signal level and parity checks are helpful clues, they are not enough. A single decoder still can't always make correct decisions on the transmitted bits and often will come up

with a wrong string of bits—the decoder is lost in a universe of codewords, and the codeword it chooses as the decoded data is not always the right one. That's why a decoder alone can't do the job.

But it turns out that the reliability information of one decoder is useful to the other and vice versa, because the two strings of parity bits refer to the very same data; it's just that the bits are arranged in a different order. So the two decoders are trying to solve the same problem but looking at it from different perspectives.

The two decoders, then, can exchange reliability information in an iterative way to improve their own decoding. All they have to do, before swapping reliability strings, is arrange the strings' content in the order each decoder needs. So a bit that was strongly detected as a 1 in one decoder, for example, influences the other decoder's opinion on the corresponding bit.

In the rain analogy, imagine you see a colleague going outside carrying an umbrella. It's a valuable additional piece of information that would affect your guess. In the case of the turbo decoders, now each decoder not only has its own "opinion," it also has an "external opinion" to help it come up with a decision about each bit. "It's as if a genie had given you that information," says Gerhard Kramer, a researcher in the mathematical sciences research center at Bell Labs. This genie whispers in your ear how confident you should be about a bit's being a 1 or a 0, he says, and that helps you decode that bit.

At the heart of turbo coding is this iterative process, in which each component decoder takes advantage of the work of the other at a previous decoding step. After a certain number of iterations, typically four to 10, both decoders begin to agree on all bits. That means the decoders are not lost anymore in a universe of codewords; they have overcome the complexity barrier.

"It's a divide-and-conquer solution," says Robert J. McEliece, a professor of electrical engineering at the California Institute of Technology, in Pasadena, and an IEEE Fellow. "It broke the problem into two smaller pieces, solved the pieces, and then put the pieces back together."

Another way of thinking about the turbo decoding process is in terms of crossword puzzles, Berrou says. Imagine that Alice solved a crossword and wanted to send the solution to Bob. Over a noiseless channel, it would be enough to send the array with the words. But over a noisy channel, the letters in the array are messed up by noise. When Bob receives the crossword, many words don't make sense. To help Bob correct the errors, Alice can send him the clues for the horizontal and vertical words. This is redundant information, since the crossword is already solved, but it nevertheless helps Bob, because, as with parity bits, it imposes constraints on the words that can be put into the array. It's a problem with two dimensions: solving the rows helps to solve the columns and vice versa, like one decoder helping the other in the turbo-decoding scheme.

Flash back 11 years as an amused 42-year-old Berrou wanders the corridors of the convention center in Geneva, peeking over the shoulders of other attendees and seeing many of them trying to understand his paper. At the presentation, young Ph.D. students and a scattering of coding veterans pack the auditorium, with people standing by the door. When Berrou and Glavieux finish, many surround them to request more explanations or simply to shake their hands.

Still, convincing the skeptics that the work had no giant overlooked error took time. "Because the foundation of digital communications relied on potent mathematical considerations," Berrou recollected later, "error-correcting codes were believed to belong solely to the world of mathematics."

What led Berrou and Glavieux to their important breakthrough was not some esoteric theorem but the struggle to solve real-world problems in telecommunications. In the late 1980s, when they began to work on coding schemes, they were surprised that an important concept in electronics—feedback—was not used in digital receivers.

In amplifiers, a sample of the output signal is routinely fed back to the input to ensure stable performance. Berrou and Glavieux wondered, why shouldn't it work for coding as well?

They ran the first experiments with their novel coding scheme in 1991 using computer simulations, and when the results came out, they were stunned. "Every day I asked myself about the possible errors in the program," says Berrou.

The first thing Berrou and Glavieux did after confirming that their results were correct was to patent the invention in France, Europe, and the United States. At the time, France Télécom was the major sponsor of their work, so the French company took possession of the turbo code patents. The inventors and their institution, however, share part of the licensing profits. (Turbo codes were not patented in Asia, where they can therefore be used for free.)

It was France Télécom that asked Berrou to come up with a commercial name for the invention. He found the name when one day, watching a car race on TV, he noticed that the newly invented code used the output of the decoders to improve the decoding process, much as a turbocharger uses its exhaust to force air into the engine and boost combustion. Voilà: "turbo codes"!

Turbo codes are already in use in Japan, where they have been incorporated into the standards for third-generation mobile phone systems, known officially as the Universal Mobile Telecommunications System (UMTS). Turbo codes are used for pictures, video, and mail transmissions, says Hirohito Suda, director of the Radio Signal Processing Laboratory at NTT DoCoMo, in Yokosuka, Japan. For voice transmission, however, convolutional codes are used, because their decoding delays are smaller than those of turbo codes.

In fact, the decoding delay—the time it takes to decode the data—is a major drawback to turbo codes. The several iterations required by turbo decoding make the delay unacceptable for real-time voice communications and other applications that require instant data processing, like hard disk storage and optical transmission.

For systems that can tolerate decoding delays, like deep-space communications, turbo codes have become an attractive option. In fact, last September, the European Space Agency, based in Paris, France, launched SMART-1, the first probe to go into space with data transmission powered by turbo codes. ESA will also use the codes on other missions, such as Rosetta, scheduled for launch early this year to rendezvous with a comet. The National Aeronautics and Space Administration, in Washington, D.C., is also planning missions that will depend on turbo codes to boost reliable communications. "The first missions that will be using these codes will be Mars Reconnaissance Orbiter and Messenger," says Fabrizio Pollara, deputy manager of the communications systems and research section at NASA's Jet Propulsion Laboratory in Pasadena, Calif.

Beyond error correction, turbo codes are helping Mobile devices achieve better reception

Digital audio broadcasting, which provides CD-quality radio programs, and satellite links, such as the new Global Area Network of Inmarsat Ltd., in London, are both also about to incorporate turbo codes into their systems.

And beyond error correction, turbo codes—or the so-called turbo principle—are also helping engineers solve a number of communications problems. "The turbo-coding idea sparked lots of other ideas," says Lajos Hanzo, a professor in the School of Electronics and Computer Science at the University of Southampton, United Kingdom, and an IEEE Fellow. One example is in trying to mitigate the effects of multipath propagation—that is, signal distortion that occurs when you receive multiple replicas of a signal that bounced off different surfaces. Turbo codes may eventually help portable devices solve this major limitation of mobile telephony.

Finally, another major impact of turbo codes has been to make researchers realize that other capacity-approaching codes existed. In fact, an alternative that has been given a new lease on life is low-density parity check (LDPC) codes, invented in the early 1960s by Robert Gallager at MIT but largely forgotten since then. "In the 1960s and 1970s, there was a very good reason why nobody paid any attention to LDPC codes," says MIT's Forney. "They were clearly far too complicated for the technology of the time."

Like turbo codes, LDPC attains capacity by means of an iterative decoding process, but these codes are considerably different from turbo codes. Now researchers have implemented LDPC codes so that they actually outperform turbo codes and get even closer to the Shannon limit. Indeed, they might prove a serious competitor to turbo codes, especially for next-generation wireless network standards, like IEEE 802.11 and IEEE 802.16. "LDPC codes are using many of the same general ideas [as turbo codes]," says Caltech's McEliece. "But in certain ways, they are even easier to analyze and easier to implement." Another advantage, perhaps the biggest of all, is that the LDPC patents have expired, so companies can use them without having to pay for intellectual-property rights.

Turbo codes put an end to a search that lasted for more than 40 years. "It's remarkable, because there's this revolution, and nowadays if you can't get close to Shannon capacity, what's wrong with you?" says the University of Illinois's Tanner. "Anybody can get close to the Shannon capacity, but let's talk about how much faster your code goes...and if you are 0.1 dB from Shannon or 0.001 dB."

It was the insight and naiveté typical of outsiders that helped Berrou and Glavieux realize what the coding theory community was missing. "Turbo codes are the result of an empirical, painstaking construction of a global coding/decoding scheme, using existing bricks that had never been put together in this way before," they wrote a few years ago.

Berrou says their work is proof that it is not always necessary to know about theoretical limits to be able to reach them. "To recall a famous joke, at least in France," he says, "the simpleton didn't know the task was impossible, so he did it."

TO PROBE FURTHER

The 2004 International Conference on Communications, to be held in Paris on 2024 June, will include several sessions on turbo codes. See <http://web.archive.org/web/20050309233014/http://www.icc2004.org/>.

"What a Wonderful Turbo World," an electronic book by Adrian Barbulescu, contains a detailed analysis of turbo codes and source code in C for simulations. See <http://web.archive.org/web/20050309233014/http://people.myoffice.net.au/~abarbulescu/>.

For a discussion of implementation issues and a presentation of a real-life prototype, see *Turbo Codes: Desirable and Designable*, by A. Giulietti, B. Bougard, and L. Van der Perre (Kluwer Academic, Dordrecht, the Netherlands, 2004).

[Home](#) | [Search](#) | [Table of Contents](#) | [IEEE Job Site](#) | [Advertising](#) | [Top](#)

1

>> **IEEE
Spec
Adv
Mai**

**The fi
con
inform
here**

What's P
Third Dig
Revoluti
Find out,
issue of 1
Forum Jo
analyst A

Prototype
Boards fr
PCBexp
Leading
supplier
circuit bo
Successf
pcbs...

PCBpro-
Quote/O
Boards
Free quo
boards ir
no sign u
Easy ord

Astec Po
Leader ir
DC-DC F
Downloa
and use
Wizard to
power sc

New Acq
Digitizers
GS/s
Acqiris u
bit digitiz
sampling
8 GS/s. 9

Buy a I



[Copyright](#) | [Terms & Conditions](#) | [Privacy & Security](#) | [Subscription Pro](#)



URL: <http://www.spectrum.ieee.org> (Modified: 27 February 2004)